JOB SUBMISSION ON GRID

An Users Introduction

Rudra Banerjee
ADVANCED COMPUTING LAB.
Dept. of Physics, University of Pune

February 2, 2008

Contents

preface **iv**

1 The Garuda Grid **1**
1.1 What is Grid??? 1
1.2 Garuda Grid 2

2 Prerequisites **5**
2.1 Getting Permission 5
2.2 Do You Have the Right BINARY File? 5
2.3 Creating binaries in remote machine 6

3 A Little on RSL File **7**
3.1 Syntax of RSL Files 8
3.2 Other Commands 9

4 A Little on Grid Variables GRAM, GASS and all that **11**
4.1 GRAM 11
4.2 GASS 12

5 Submitting Jobs **13**
5.1 globusrun 13
5.2 globus-job-run 14
5.3 globus-job-submit 15

A The Garuda Portal: How Should You Use It **17**

preface

This is an user's introduction to grid using Globus®Toolkit from an user's point of view. With a brief introduction to what grid is, I have shifted quickly to the game itself. In this part, i have done an step by step introduction, starting from the access to the grid to submitting the job.

In the appendix, a special note is there on using GARUDA grid.

Hope this will be of help for the users.

Chapter 1

The Garuda Grid

1.1 What is Grid???

A computational Grid is an aggregation of heterogeneous and geographically distributed resources such as computing, storage and special equipments. The aim of the grid is to dynamically share resources to facilitate optimal utilization. Grid allows us to solve complex problems that could not be dealt with otherwise. It provides high-speed and seamless access to supercomputers and also enables creation and management of dynamic virtual collaborations and organizations

Leading scientist is this field, Dr. I. Foster has given *Three Point Checklist* in this purpose :

A Grid is a system that

- coordinates resources that are not subject to centralized control

  Integrates and coordinates different resources and users that live in different control domain

- using standard, open, general-purpose protocols

  Protocols such as authentication, authorization and access should be

open and standard and open. Otherwise the system is application specific.and

- interfaces to deliver nontrivial qualities of service
  Should deliver various quality of service in a coordinated way so that the utility of the system is **significantly** greater then the sum of its part.

and is accepted as present standard.

1.2 Garuda Grid

GARUDA is a collaboration of science researchers and experimenters on a nation wide grid of computational nodes, mass storage and scientific instruments that aims to provide the technological advances required to enable data and compute intensive science for the 21st century.

GARUDA aims at strengthening and advancing scientific and technological excellence in the area of Grid and Peer-to-Peer technologies. To achieve its objective, GARUDA brings together a critical mass of well-established researchers, from 48 research laboratories and academic institutions, who have constructed an ambitious program of activities. (35 Research and Academic Institutions, 11 C-DAC centers and ERNET, Delhi). Right now four nodes of this Grid are working- Pune, Bangalore, Chennai and Hyderabad center of C-DAC. To provide high performance computing facilities and services for the scientific and research community and for the enterprise, C-DAC had earlier set up the National PARAM Supercomputing Facility (NPSF) at Pune, housing its PARAM 10000 supercomputer with a peak computing power of 10 Gflops. In its continuing effort in this direction, C-DAC has established Center for Tera-scale Supercomputing Facility (CTSF) at its C-DAC Knowledge Park, Bangalore with the PARAM Padma machine in its Tera-scale configuration. The PARAM Padma at CTSF has a peak computing power of one Tera-flop.

In this four center, not all of the machines are equivalent. Like Pune is Linux cluster, on the other hand B'lore has Linux, AIX and Solaris. There main capacities and IP are given in the table:

GARUDA GRID IN BRIEF

place	resource	number	IP
Pune	Linux Cluster	16+head node	xn00.npsf.cdac.ernet.in
B'lore	Linux (Xeon) Cluster	5 nodes(10 CPU)	xn02.ctsf.cdac.org.in
	AIX Cluster		tf34.ctsf.cdac.org.in
	Solaris Cluster		e1.ctsf.cdac.org.in
H'bad	Linux Cluster	5 nodes(10 CPU)	hyd01.hardware.cdac.ernet.in
Chennai	Linux Cluster		che01.hardware.cdac.ernet.in

Resources available in the Grid

Total number of Nodes = 10

Node Name	Head Node	OS	ARCH	Load Average	Status	Conf Procs	Available Procs
xn02.ctsf.cdac.org.in	xn02	linux	linux	8.970000	Idle	2	0
xn01.ctsf.cdac.org.in	xn02	linux	xeon	5.000000	Down	2	0
xn03.ctsf.cdac.org.in	xn02	linux	linux	9.000000	Idle	2	0
xn04.ctsf.cdac.org.in	xn02	linux	linux	54.000000	Idle	2	0
gsiftp://xn02/tmp/	SHARED	-	-	-	Idle	-	0
tf39.ctsf.cdac.org.in	tf35	AIX51	R6000	-	Idle	4	4
tf36.ctsf.cdac.org.in	tf35	AIX51	R6000	-	Idle	4	4
tf35.ctsf.cdac.org.in	tf35	AIX51	R6000	0.017761	Idle	4	4
tf29.ctsf.cdac.org.in	tf35	AIX51	R6000	0.061829	Idle	4	4
lsfhost.localdomain	n24	-	-	-	Idle	34	34

Legend

```
'-' represents information unavailable
'Conf Procs' represents Configured Number of Processors
'Available Procs' represents Available Number of Processors
```

Status	Description
Idle	All the processors in the node are available for Job Submission
Running	Partial number of processors in the node are available for Job Submission
Busy	All the processors in the node are in use
Down	None of the processors in the node are up

Figure 1.1: Grid resources: screen-shot

Chapter 2

Prerequisites

2.1 Getting Permission

First thing first. To submit job in a grid you must have to have a login and fully working account in the grid-client's node. Next thing is you must have the *pass phrase* for the grid-proxy to submit jobs in grid. Once you submit grid-proxy-init you will get the access of the grid, for 1 day (24 hr)by default. It is advisable to get a tentative idea about the running time of your job and if it is more then 1 day, ask permission for longer period. Some often used option of grid-proxy-init are given below:

Syntax: grid-proxy-init [-help][-pwstdin][-limited][-hours H] ...

Options

-help, -usage	Displays usage
-verify	Verifies certificate to make proxy for
-pwstdin	Allows passphrase from stdin
-hours H	Proxy is valid for H hours (default:12)

2.2 Do You Have the Right BINARY File?

Given that the resource is varied, it is assumed that the user should have a matching binary. For example, if your binary is compiled on Intel machine,

it is most suitable for Xeons; but it is likely that it may not run on Solaris, as the Fortarn90 compiler of Solaris is not dependable at all[1] .

Now few more things you should remember. If you are using a binary that calls other library file (more often then not, they do) you *static*ally compiled version of the binary. For Intel fortran, the command is

```
$ ifort -static my_file.f90
```

2.3 Creating binaries in remote machine

The most important thing in using grid is making the right binary files for right architecture. Suppose you want to make a binary for AIX or Solaris. If you have access to these locally, then there is no problem. But if you dont have, you have to create it. There's nothing you can do in the RSL to submit a *.f (or any other) and have it be automatically compiled.

Your main choice (depending on how expensive time on the Solaris machine is) is to just submit a job to compile the .f file and stage out the executable, then submit that executable. If you were confident about compiling it, you could also just submit a shell script that compiled the program as step one, then ran it as step two.(I would prefer it, as it seems more logical to me and also a two line script if you are familier with script.)

Otherwise, what you're looking for sounds like cross-compilation. There was a short thread about it on the gfortran mailing list recently: http://gcc.gnu.org/ml/fortran/2006-11/msg00517.html. Basically you can setup your fortran compiler to emit Solaris executables, but it sounds like more work than just compiling it directly.

[1] I ll advice to avoid f90 jobs in Solaris as the fortran 90 compiler (CF90)of this system is not dependable.

Chapter 3

A Little on RSL File

The Globus Resource Specification Language (RSL) provides a neutral way to describe the resource needs of submitted jobs, a way that can be used by other parts of the Globus job-management system and that can be translated into each participating computer's local job-management jargon . The Globus Resource Allocation Manager (*GRAM*) relies on RSL strings to perform its management functions and to coordinate with other Globus software.

RSL provides

1. a syntax used to compose complex resource descriptions from basic elements,

2. a way to record resource decisions as attribute-value pairs for later reuse, and

3. a vocabulary of job attributes, each of which serves as a parameter to control job behavior when passed to the (remote) system that actually schedules the job.

The Globus web site offers several detailed descriptions of and technical specifications for RSL. This subsection (and the next two) summarize the most important RSL features and the aspects of RSL most relevant for remote users submitting jobs to LC machines

3.1 Syntax of RSL Files

The first recognizer is about job numbers:
& begins all single-job (*GRAM*-managed) resource descriptions.
+ begins all multiple-job (DUROC-managed) resource descriptions. The next thing is different attributes:
(attribute = "value")
assigns value to the job resource named by attribute. Each attribute/value pair is called an RSL relation, and each is separately enclosed in parentheses. On the very first, we generally mention executable. For my PWscf submission, I have

Example
(executable=$(GLOBUSRUN_GASS_URL)/home/garuda/rudra/pw/espresso-3.1/bin/pw.x)

□RB⌋

The next most important attributes are *stdin, stdout* and *stderr*. They informs *GRAM* about the input, output and the error file. In my above stated PWscf submission, these three reference are

Example
(stdin=$(GLOBUSRUN_GASS_URL)/home/garuda/rudra/al.in)
(stdout=$(GLOBUSRUN_GASS_URL)/home/garuda/rudra/pwout)
(stderr=$(GLOBUSRUN_GASS_URL)/home/garuda/rudra/pwerr)

□RB⌋

In above examples, my executable is *pw.x* which is in the directory */home/garuda/rudra/pw/espresso-3.1/bin/*

my input is *al.in* in */home/garuda/rudra/* and I expect the outputs *stdout & stderr* to be in in the same directory.

The next attribute one need is *arguments*. This is used in various purpose from reading input values to shifting library files to working site, depending on the argument.

To run PWscf, you need to have your Pseudopotential files in the working node. There is to way to do that. one is sftp it manually (which is tedious when you need a lot of such file); and the other is to use *arguments* to do this job. So f my Pseudopotential files are in the directory */home/garuda/rudra/pw/espresso-3.1/Pseudo/* and I need the file *Al.vbc.UPF* then the *argument* looks like

Example
(arguments=file:///home/garuda/rudra/pw/espresso-3.1/Pseudo/Al.vbc.UPF gsiftp://remote_pwd/Al.vbc.UPF)

□RB⌋

remote pwd is the slot where your job is supposed to run. This is the full *rsl* file, let it be named as **pw.rsl**

3.2 Other Commands

According to your need, you may need many other commands to submit jobs successfully and comfortably. One of the is already shown in argument. There my file */pw/espresso-3.1/Pseudo/Al.vbc.UPF* has to be shifted to working node of the grid. *gsigtp* is used to do this job. If the situation is such that you have to move your file to a third location, then you can use gsiftp twice. Suppose you want to move a file from working site to a third site. Then you should use the command

Example
(arguments=gsiftp://working node/path/target file
gsiftp://remote site/path/destination)

□RB⌋

globus-url-copy or *gridftp* are two other command used for same job.

Chapter 4

A Little on Grid Variables

GRAM, GASS and all that

4.1 GRAM

The Globus Toolkit includes a set of service components collectively referred to as the *Grid Resource Allocation and Management* (**GRAM**). *GRAM* simplifies the use of remote systems by providing a single standard interface for requesting and using remote system resources for the execution of "jobs". The most common use (and the best supported use) of *GRAM* is remote job submission and control. This is typically used to support distributed computing applications. *GRAM* is designed to provide a single common protocol for requesting and using remote system resources, by providing a uniform, flexible interface to local job scheduling systems. *GRAM* reduces the number of mechanisms required for using remote resources (such as remote compute systems). Local systems may use a wide variety of management mechanisms (schedulers, queuing systems, reservation systems, and control interfaces), but users and application developers need to learn how to use only one *GRAM* to request and use these resources. This connection is maintained through *GRAM*. Both sides need work only with *GRAM*, so the number of interactions and protocols that need to be used are greatly reduced. *GRAM* does

not provide scheduling or resource brokering capabilities.

4.2 GASS

Global Access to Secondary Storage (**GASS**) simplifies the porting and running of applications that use file I/O to the Globus environment. Libraries and utilities are provided to eliminate the need to

1. manually login to sites and ftp files
2. install a distributed file system

The APIs are designed to allow reuse of programs that use Unix or standard C I/O with little or no modification. Currently the ftp and x-gass (GASS server) protocols are supported. The typical user's view of GASS comprises a simple file access API for opening, closing, and prefetching files. Planned are some RSL extensions for remote cache management.

1. File Access API (to be defined).
2. RSL extensions.

Chapter 5

Submitting Jobs

So, now you know a lot about grid terminology and ready to submit (at least I hope) jobs on Grid. Lets have a look on your resources. You have a binary executable and inputs to run that file, right? So go for shooting the job. There are several options for shooting jobs on Grid.

5.1 globusrun

The easiest submission is *globusrun*. The **.rsl* file is used here and I think it is best to start with that.

Example

```
$globusrun -s -r abcd -f pw.rsl
```

□RB⌋

where *abcd* is the remote machine where I want to fire the job. Check the table at Ch. 1 for actual host name.

The globusrun syntax is as follows:

Syntax:

globusrun [options] [RSL String]

-help|	-usage
	Display help
-i |	-interactive
	Run globusrun in interactive mode (multi requests only)
-f<rsl filename> |	-file <rsl filename>
	Read RSL from the local file <rsl filename>. The RSL can be either a single job request, or a multi request
-q |	-quiet
	Quiet mode (do not print diagnostic messages)
-o |	-output-enable
	Use the GASS Server library to redirect standout output and standard error to globusrun. Implies -quiet
-s |	-server
	$(GLOBUSRUN_GASS_URL) can be used to access files local to the submission machine via GASS. Implies -output-enable and -quiet
-w |	-write-allow
	Enable the GASS Server library and allow writing to GASS URLs. Implies -server and -quiet.

5.2 globus-job-run

globus-job-run is an on line interface to job submission, featuring staging of data and executables using a GASS server. Staging is the process in which files are copied from the local machine to the remote machine on which the processing takes place and automatically deleted from it when processing is complete. This command is used for short, interactive jobs in foreground.

The basic syntax is

Example

$globus-job-run work-site -s binary arg

□RB⌋

For multiple submission, it looks like:

Example

$ globus-job-run

$ -: host1 np 2 s myprog.linux arg1

$ -: host2 np 3 s myprog.aix arg2

□RB⌋

where *host* is the remote work node, *-np n* is the number of processors you want for a job called *myprog.** . Here you can see that since I am using same program but compiled in different system(Linux and AIX) I am supposed to choose my host accordingly.
Other frequently used options are *stdin,stderr & stdout*. Job stdin defaults to */dev/null*.
For other commands, go for
$globus-job-run -help

5.3 globus-job-submit

This is little better then globus-job-run in the sense that that it has a batch interface.

The job is submitted using globus-job-submit , its status is then checked using globus-job-status , the standard output of the job is the obtained by

using globus-job-get-output and then the job is finally cleaned using the globus-job-clean command.

Appendix A

The Garuda Portal: How Should You Use It

There is two way of submitting jobs is Garuda grid. One is what I talked about...commandline submission; and the second is via Portal. You can access this only if you are log in to the grid http://192.168.60.40:8080/GridPortal/.

But even you prefer to use command line submission, then also the Portal is very helpful. Most important of them is monitor the workload on different noads. By this you can choose where to shoot the jobs. The page shown, gives many first hand information like CPU load and noad status. From then they give the exact status (Like of there available resources, status,Installed packages)of different noad. It is strongly advised to keep a look on the state of a specific node before shooting jobs there.

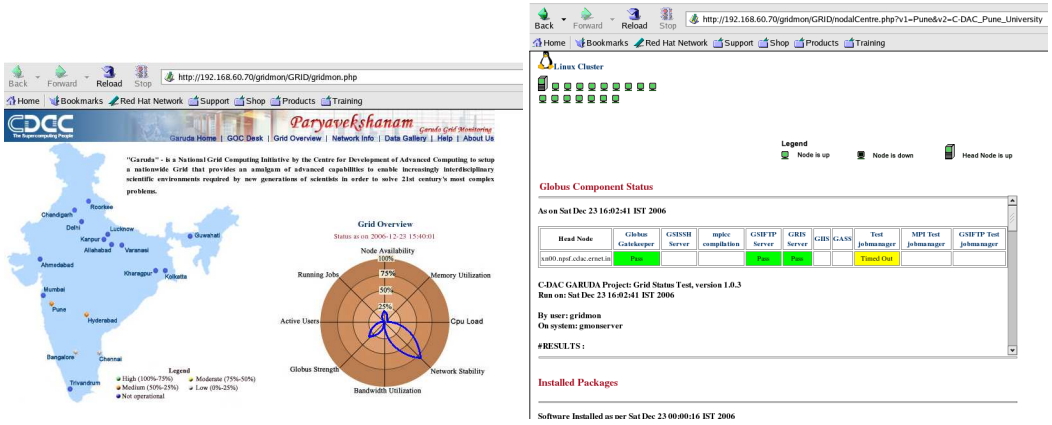

Figure A.1: Screen-shot: Garuda Portal